INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

An automatic correcting method

Philippe Langlois, INRIA

No 4204

juin 2001

THÈME 2

ISSN 0249-6399 ISRN INRIA/RR--4204--FR+ENG

RHÔNE-ALPES

An automatic correcting method

Philippe Langlois, INRIA

Thème 2 — Génie logiciel
et calcul symbolique
Projet Arénaire

Rapport de recherche n ° 4204 — juin 2001 — 14 pages

Abstract: Correcting methods intend to improve the accuracy of results computed in finite precision. The CENA method processes an automatic correction of the first-order effect of the rounding errors the computation generates. The method provides a corrected result and a bound of the residual error for a class of algorithms we identify. We present the main features of the CENA method and illustrate its interests and limitations with examples.

Key-words: automatic correction, CENA method, accuracy, finite precision, floating point arithmetic

(Résumé : tsvp)

This text is also available as a research report of the Laboratoire de l'Informatique du Parallélisme `http://www.ens-lyon.fr/LIP`.

Unité de recherche INRIA Rhône-Alpes
655, avenue de l'Europe, 38330 MONTBONNOT ST MARTIN (France)
Téléphone : 04 76 61 52 00 - International: +33 4 76 61 52 00
Télécopie : 04 76 61 52 52 - International: +33 4 76 61 52 52

Une méthode de correction automatique

Résumé : Les méthodes de correction améliorent la précision des résultats calculés en précision finie. La méthode CENA permet une correction automatique de l'effet d'ordre un des erreurs d'arrondi introduites par le calcul. Elle fournit un résultat corrigé et une borne de l'erreur résiduelle pour une classe d'algorithmes aux propriétés identifiées. Nous présentons les principales caractéristiques de la méthode CENA et illustrons ses intérêts et ses limitations par des exemples.

Mots-clé : correction automatique, méthode CENA, précision, précision finie, arithmétique flottante

1 Corriger : pourquoi et comment

L'arithmétique des ordinateurs est une approximation de l'arithmétique réelle qui s'en distingue par *la précision finie* des nombres qu'elle manipule. Le résultat approché fourni par un calcul en précision finie représente en général le résultat exact à *une précision* qui est au mieux celle de l'arithmétique utilisée.

1.1 Objectif : améliorer la précision du résultat

Remarquons les deux sens différents de la notion de précision telle que nous venons de l'utiliser.
La précision finie de l'arithmétique. La précision finie de l'arithmétique est une quantité arbitrairement petite mais fixée *a priori*. Dans la pratique, cette précision est une fonction de caractéristiques matérielles comme par exemple le nombre de bits de la mantisse binaire d'un nombre à virgule flottante et le mode d'arrondi. Notons $\mathbf{u}$ cette précision finie. Nous la supposerons définie de façon relative car nous nous intéressons plus particulièrement aux nombres à virgule flottante, ou nombres flottants ; on a ainsi $\mathbf{u} < 1$. Soit $\mathbb{F}$ un ensemble de nombres flottants où $fl(x) \in \mathbb{F}$ dénote l'arrondi dans $\mathbb{F}$ de la valeur $x \in \mathbb{R}$. L'ensemble $\mathbb{F}$ est muni d'une arithmétique à précision finie $\mathbf{u}$ dés lors que $fl(x \circ y)$, l'évaluation sur $\mathbb{F}$ de l'opérateur élémentaire $\circ \in \{+, -, \times, /, \sqrt{}\}$ vérifie pour $x, y \in \mathbb{F}$,

$$fl(x \circ y) = (x \circ y)(1 + \delta) \quad \text{avec} \quad |\delta| \leq \mathbf{u}, \tag{1}$$

en supposant qu'aucun dépassement de capacité (*overflow, underflow*) n'apparaisse lors du calcul. Dans le cas de la norme d'arithmétique flottante binaire IEEE-754, deux formats de représentation définissent respectivement les simple et double précisions $\mathbf{u}_s \approx 5.96 \times 10^{-8}$ et $\mathbf{u}_d \approx 1.11 \times 10^{-16}$ en mode d'arrondi au plus près [13]. Si on suppose de plus que $\mathbf{u} \in \mathbb{F}$, la relation (1) permet d'expliciter la précision $\mathbf{u}$ pour l'arrondi au plus près,

$$\mathbf{u} = \sup\{x \in \mathbb{F} : fl(1 + x) = 1\}. \tag{2}$$

La précision d'un résultat. Le second sens de précision décrit l'erreur qui existe entre la quantité réelle x et la quantité calculée $\widehat{x}$ qu'elle approche. Parce qu'il s'agit d'une relation entre valeurs exacte et approchée, cette notion de précision n'est pas fixée *a priori*, elle dépend de l'algorithme de calcul de $\widehat{x}$. Lorsque celui-ci est exécuté en précision finie $\mathbf{u}$, il est raisonnable de considérer que la précision de $\widehat{x}$ est optimale dès que

$$\widehat{x} = x(1 + \delta) \quad \text{avec} \quad |\delta| \leq \mathbf{u}. \tag{3}$$

Cette relation étend à l'algorithme de calcul de $\widehat{x}$, la contrainte de précision (1) satisfaite par les opérations élémentaires sur $\mathbb{F}$.

Il est facile de se convaincre qu'une telle contrainte n'est plus nécessairement satisfaite lorsque l'algorithme qui calcule $\widehat{x}$ évalue plusieurs opérations élémentaires. Peu d'opérations suffisent à dégrader la précision d'un résultat comme par exemple pour la somme suivante où nous notons $\overline{\mathbf{u}} = \mathbf{u}^{-1}$ et supposons $\overline{\mathbf{u}} \in \mathbb{F}$. Le calcul de $y = \overline{\mathbf{u}} + 1$, puis $x = \overline{\mathbf{u}} - y$ donne en précision $\mathbf{u}$,

$$fl(y) = \overline{\mathbf{u}},$$

donc

$$fl(x) = 0,$$

tandis que

$$x = 1.$$

Cet algorithme de calcul de $\widehat{x}$ ne satisfait pas la relation (3). Il est immédiat ici de proposer un algorithme qui calcule $\widehat{x} = x$. La relation (3) est alors satisfaite avec $\delta = 0$ et le résultat maintenant calculé est même ici le résultat exact.

1.2 Comment améliorer la précision du résultat

L'exemple précédent suggère plusieurs pistes pour améliorer la précision d'un résultat calculé en précision finie.

Augmenter la précision du calcul. La plus naïve consiste à effectuer le même calcul avec une arithmétique d'une précision supérieure. La perte de précision de l'exemple précédent est atténuée ou disparaît en précision augmentée $\mathbf{u}_e < \mathbf{u}$.

Cette précision supérieure peut être disponible de façon matérielle comme par exemple la double précision de la norme IEEE-754 si le calcul initial est en simple précision. Au-delà, la précision étendue de la double précision de la norme IEEE-754 fournit un cadre normatif à différentes réalisations, à la fois matérielle et logicielle, dont la précision effective varie selon les architectures. Sur processeurs SPARC, l'implantation logicielle d'une véritable quadruple précision est disponible, elle permet $\mathbf{u} \approx \mathbf{u}_d^2$. Sur processeurs x86, la précision étendue permet plus modestement $\mathbf{u} \approx \mathbf{u}_d^{1.2}$.

Des alternatives logicielles qui permettent une précision arbitrairement élevée existent ; on parle alors de multi-précision. Les bibliothèques de multi-précision sont nombreuses et diffèrent selon qu'elles représentent ces nombres de précision arbitraire avec une mantisse arbitrairement longue (et un exposant fixé) ou qu'elles les décomposent en plusieurs champs d'exposants différents. Parce qu'elles peuvent utiliser les opérateurs flottants sur chaque champ, ces dernières ont les performances les plus intéressantes (mais qui restent malgré tout très faibles). Nous renvoyons le lecteur à [26], plus loin dans ce volume, pour une discussion plus exhaustive de ces nombreuses variantes.

Comme la précision arbitraire génère un sur-coût élevé, certaines bibliothèques minimisent les pertes de performances par rapport aux précisions classiques en fixant un format de précision étendue *a priori*. Les principales réalisations sont les bibliothèques *double-double* de Bailey [1] ou de Briggs [4], ou les *quad-double* de Bailey *et al.* [11]. Elles permettent respectivement une précision de l'ordre de $\mathbf{u}_d^2$ ou $\mathbf{u}_d^4$ en représentant un flottant en précision étendue par la somme non évaluée d'une paire ou d'un quadruplet de flottants en double précision. En ce qui concerne les performances de ces bibliothèques, Briggs signale pour les *double-double* un facteur pénalisant variant de 10 à 25 par rapport à la double précision IEEE [4] ; un facteur d'au moins 100 ressort des mesures proposées pour les *quad-double* dans [11].

Réécrire les algorithmes. Une simple modification de l'ordre des instructions de l'algorithme de l'exemple précédent conduisait au résultat exact. Réécrire les algorithmes peut permettre de réduire l'erreur finale. Hélas, une telle approche n'est pas suffisamment générale car l'ordre optimal des instructions dépend très souvent des valeurs des données. De plus, rien ne garantit que, bien qu'améliorée, la précision de la solution calculée soit suffisante.

La sommation de n scalaires est l'exemple classique de ces deux aspects. Les algorithmes de sommation sont nombreux et la plupart ne diffèrent que par l'ordre des additions successives. C'est le cas de la sommation récursive, non ordonnée ou ordonnée par valeurs absolues croissantes ou décroissantes, de la sommation par paire [18] ou de la sommation par insertions [23]. D'autres algorithmes introduisent de nouvelles quantités ; nous les considérons dans le paragraphe suivant. Sans hypothèse particulière sur les données, aucun des algorithmes cités ne fournira systématiquement une solution plus précise que les autres. Bien sûr, si les n opérandes sont par exemple toutes positives, la sommation récursive par ordre croissant minimise l'erreur dans le résultat calculé. Cependant, rien n'assure que le résultat calculé est une approximation précise de la somme exacte, ni qu'un autre algorithme ne peut fournir un résultat plus précis [12].

Corriger les algorithmes. L'exemple de la sommation de n scalaires permet d'illustrer l'amélioration de la précision des résultats par correction, approche qui nous intéresse plus particulièrement dans cet article. Ces algorithmes introduisent d'autres quantités que les opérandes. Une telle approche est possible grâce au résultat suivant que l'on énonce dans l'ensemble $\mathbb{F}$ des flottants binaires muni d'une arithmétique avec arrondi au plus prés.

Théorème 1 (Dekker, 1971) *Soient a et b deux flottants de $\mathbb{F}$ tels que $|a| > |b|$ et $\widehat{s} = fl(a+b)$. On a*

$$a + b = \widehat{s} + \delta,$$

où δ est le flottant de $\mathbb{F}$ qui se calcule par la relation $\delta = fl(fl(a - \widehat{s}) + b)$.

Ce théorème prouve que l'erreur d'arrondi d'une addition de deux flottants est elle-même un flottant et qu'elle est de plus calculable exactement à l'aide de l'addition flottante. Kahan utilise ce résultat pour corriger la

somme de n scalaires x_i [14]. Après chaque somme partielle $s_i = s_{i-1} + x_i$, son algorithme de sommation compensée calcule l'erreur associée $\delta = \delta(i-1, i)$ et l'ajoute au terme suivant x_{i+1} avant le calcul de la somme partielle suivante s_{i+1}. Il est important de se convaincre que rajouter δ à s_i ne modifie pas s_i. Cet algorithme améliore *la borne* sur l'erreur de la somme calculée d'un facteur de l'ordre de n par rapport aux algorithmes de sommation précédemment cités (tant que $n\mathbf{u} \leq 1$). La précision de la sommation compensée ne satisfait pas encore pour autant la précision optimale de la relation (3).

De nombreuses variations de ce type de correction ont été depuis proposées. Certaines accumulent les erreurs intermédiaires pour appliquer une correction finale une fois ou de façon itérative [19]. L'amélioration la plus significative est l'algorithme de sommation doublement compensée proposé 25 ans plus tard par Priest dans sa thèse [21] — dirigée par Kahan. La double compensation permet cette fois une précision presque optimale ; la somme calculée $\widehat{s}$ vérifie $|\widehat{s} - s|/|s| \leq 2\mathbf{u}$ (tant que $n\mathbf{u} \leq 8$).

Cette quête de la précision optimale induit un coût non négligeable. Les algorithmes de sommation compensée ou doublement compensée ajoutent respectivement 3 et 9 additions à chaque somme partielle.

La méthode de correction automatique que nous présentons dans la section suivante généralise à d'autres algorithmes que la seule sommation, ce principe de calculer un terme correctif à partir des erreurs intermédiaires.

2 Principes de la méthode CENA

La méthode de correction automatique CENA consiste à calculer un terme correctif linéaire et une majoration des erreurs introduites par le calcul en précision finie de cette correction. Plaçons nous dans l'ensemble des flottants $\mathbb{F}$ de précision $\mathbf{u}$. Les opérations arithmétiques élémentaires $(+, -, \times, /, \sqrt{\ })$ sur $\mathbb{F}$ vérifient la relation (1), $fl\,(x \circ y) = (x \circ y)(1+\delta)$ avec $|\delta| \leq \mathbf{u}$. Nous utilisons la notation classique $\widehat{z} = fl\,(z)$ pour les valeurs flottantes $\widehat{z}$. Présentons les principaux aspects de la méthode CENA en considérant l'évaluation numérique $\widehat{f}$ sur $\mathbb{F}$ d'une fonction f de n variables réelles. Pour simplifier, nous supposons que f est une fonction numérique réelle.

2.1 Calcul d'un terme correctif linéaire

Le calcul en précision finie $\mathbf{u}$ de $x = f(X)$ produit $\widehat{x}$ pour la donnée $X = (x_1, \ldots, x_n)$ dans $\mathbb{F}^n$. L'erreur globale directe $\widehat{x} - x$ mesure la perte de précision sur le résultat $\widehat{x}$. Le calcul de chaque variable intermédiaire $\widehat{x}_k$ introduit une erreur (absolue) élémentaire δ_k pour $k = n+1, \ldots, N$ et $\widehat{x} = \widehat{x}_N$. Pour $\widehat{x}_i$, $\widehat{x}_j$ et $\widehat{x}_k = fl\,(\widehat{x}_i \circ \widehat{x}_j)$ dans $\mathbb{F}$ telles que $1 \leq i, j < k \leq N$, et $\circ \in \{+, -, \times, /, \sqrt{\ }\}$, l'erreur élémentaire δ_k est définie par la relation

$$\widehat{x}_k = \widehat{x}_i \circ \widehat{x}_j + \delta_k = x_k + \delta_k, \tag{4}$$

où $\widehat{x}_j \neq 0$ quand $\circ = /$, et $\widehat{x}_i = 0$, $\widehat{x}_j \geq 0$ quand $\circ = \sqrt{\ }$.

L'évaluation numérique de x dans $\mathbb{F}$ s'écrit $\widehat{x} = \widehat{f}(X, \delta)$, c'est-à-dire comme une fonction de la donnée X et des erreurs élémentaires $\delta = (\delta_{n+1}, \ldots, \delta_N)$. Un approximant d'ordre 1, par rapport aux erreurs élémentaires, de l'erreur globale $\widehat{x} - x$ est alors

$$\Delta_L = \sum_{k=n+1}^{N} \frac{\partial \widehat{f}}{\partial \delta_k}(X, \delta) \cdot \delta_k \,. \tag{5}$$

Comme $x = \widehat{f}(X, 0)$, l'erreur globale directe vérifie

$$\widehat{x} - x = \Delta_L - E_L, \tag{6}$$

où E_L est *l'erreur de linéarisation* associée à Δ_L.

Il est classique d'appliquer la relation (1) pour majorer chaque erreur absolue élémentaire δ_k dans relation (5). On a alors un majorant de l'approximant $|\Delta_L|$, et en négligeant l'erreur de linéarisation E_L, on en déduit une majoration de l'erreur globale directe [7], [24]. Nous proposons ici d'utiliser des théorèmes comme le théorème 1 pour calculer un terme correctif $\widehat{\Delta}_L$ et en déduire *un résultat corrigé* $\overline{x}$ défini par

$$\overline{x} = fl\left(\widehat{x} - \widehat{\Delta}_L\right). \tag{7}$$

Nous calculons $\widehat{\Delta}_L$ en évaluant la relation (5) en arithmétique en précision finie. Les dérivées partielles sont calculées par différentiation algorithmique (ou différentiation automatique) [10]. Ces deux types de calcul — erreurs élémentaires et dérivées partielles — sont bien connus mais leur utilisation conjointe permet le calcul d'un terme correctif, ce qui constitue un des apports original de la méthode CENA.

Le résultat corrigé $\overline{x}$ est entaché de l'erreur de linéarisation E_L et d'une erreur de calcul E_C. Cette dernière est introduite par le calcul en précision finie des dérivées partielles et des erreurs élémentaires dans la relation (5), et de la correction finale (7). Ainsi, *une erreur résiduelle* sépare le résultat corrigé et le résultat exact ; elle vérifie

$$\overline{x} - x = -(E_L + E_C). \tag{8}$$

De la minimisation de cette erreur résiduelle dépend donc la précision du résultat corrigé.

2.2 Validation de la correction pour les algorithmes linéaires

Le second aspect de la méthode CENA consiste à associer au résultat corrigé $\overline{x}$ une borne de l'erreur de calcul E_C. Cette borne B_{E_C} majore l'erreur résiduelle quand $E_L = 0$. Cette dernière condition est vérifiée pour les *algorithmes linéaires*, une classe particulière que nous identifions dans cette section.

Nous avons déjà remarqué que l'ordre des opérations arithmétiques d'un algorithme influe sur la précision du résultat calculé. Les algorithmes linéaires sont tels que l'erreur globale directe $\widehat{x} - x$ est une fonction linéaire des erreurs d'arrondi élémentaires δ_k. Il est facile de vérifier que la définition suivante garantit cette propriété [16].

Définition 1 *Un* algorithme linéaire sur $\mathbb{F}$ *est un algorithme qui contient les opérations* $\{+, -, \times, /, \sqrt{}\}$ *et tel que*

- *chaque multiplication* $fl(\widehat{x}_i \times \widehat{x}_j)$ *vérifie* $\widehat{x}_i = x_i$ *ou* $\widehat{x}_j = x_j$,
- *chaque division* $fl(\widehat{x}_i / \widehat{x}_j)$ *vérifie* $\widehat{x}_j = x_j$,
- *chaque racine carrée* $fl\left(\sqrt{\widehat{x}_j}\right)$ *vérifie* $\widehat{x}_j = x_j$.

Bien que restrictive, cette définition est satisfaite par certains algorithmes de base. Citons par exemple,

- la sommation de n scalaires,
- le produit scalaire et donc la plupart des sous-programmes des BLAS (opérations entre scalaires, vecteurs et matrices),
- l'évaluation polynomiale par la méthode de Horner,
- la résolution par substitution des systèmes linéaires triangulaires.

Bien sûr, des parties d'algorithmes plus généraux peuvent satisfaire la définition 1. L'erreur globale partielle associée à ces parties, considérées comme fonctions des variables intermédiaires qui y apparaissent, est telle que $E_L = 0$ localement. L'intérêt des algorithmes linéaires pour la correction automatique (7) s'exprime par les deux résultats suivants.

Théorème 2 *Lorsque x est calculé avec un algorithme linéaire, le résultat corrigé $\overline{x}$ n'est entaché que de l'erreur de calcul E_C, c'est-à-dire*

$$\overline{x} = x - E_C.$$

Le corollaire suivant est énoncé pour un résultat scalaire x mais s'écrit de la même façon pour x quelconque dès lors que $\mathcal{I}_{E_C}$ est la boule (en norme adaptée) de centre $\overline{x}$ et de rayon B_{E_C}.

Corollaire 1 *Soit B_{E_C} un majorant de l'erreur de calcul E_C, c'est-à-dire $|E_C| \leq B_{E_C}$. Le résultat exact x et le résultat corrigé $\overline{x}$ d'un algorithme linéaire vérifient*

$$x \in \mathcal{I}_{E_C} = [\overline{x} - B_{E_C}, \overline{x} + B_{E_C}].$$

La borne B_{E_C}, qui majore les erreurs d'arrondi introduites par la correction automatique, fournit une région $\mathcal{I}_{E_C}$ autour du résultat corrigé $\overline{x}$ et le corollaire 1 garantit que le résultat exact x (inconnu) est inclus dans cette région. Cette propriété peut aussi permettre de vérifier la précision du résultat $\widehat{x}$ initialement calculé. Il faut pour cela pouvoir calculer cette borne B_{E_C} ; c'est ce que nous décrivons maintenant assez succinctement.

La majoration dynamique de l'erreur directe (*Running Error Analysis*) est une technique simple et efficace largement utilisée depuis Wilkinson [25]. Le relation (1) s'écrit aussi

$$|x \circ y - fl(x \circ y)| \leq \mathbf{u}|fl(x \circ y)|,$$

ce qui fournit un majorant de l'erreur élémentaire commise dans le calcul $fl(x \circ y)$ en fonction de quantités connues à l'exécution. Il suffit d'accumuler ces majorants élémentaires pour obtenir un majorant de l'erreur globale une fois l'algorithme exécuté. Par exemple, l'erreur introduite par la correction finale (7) satisfait

$$|e| \leq \mathbf{u}|\overline{x}|.$$

Pour ce qui est du terme correctif Δ_L, l'évaluation de la relation (5) consiste à calculer une itération du genre $D_k = D_{k-1} + d_k \times \delta_k$ $(k = n+1, \ldots, N)$. Une majoration dynamique de l'erreur introduite par cette itération s'écrit en négligeant les termes d'ordre 2 en $\mathbf{u}$,

$$|e_k| \leq |e_{k-1}| + \mathbf{u}\,(|D_k| + |d_k \times \delta_k| + \alpha_k|\delta_k| + \beta_k|d_k|).$$

Remarquons que cette expression ne dépend que de quantités calculées ou connues lors du calcul. En effet, la borne β_k majore l'erreur d'approximation de l'erreur élémentaire pour la division et la racine carrée, elle est nulle dans les autres cas. La borne α_k est obtenue par majoration dynamique de l'erreur directe de l'algorithme de différentiation algorithmique. Nous renvoyons le lecteur à [15] pour le détail des majorants β_k et un algorithme de calcul de α_k dans le cas du mode inverse de la différentiation algorithmique. La borne B_{E_C} de l'erreur introduite par le calcul de la correction s'obtient à partir des majorations précédentes,

$$B_{E_C} = \mathbf{u}(e_N + \overline{x}).$$

Bien que plus fine que les majorations *a priori*, ce type de borne dynamique souffre des limitations classiques des majorants établis dans le pire des cas. Sauf cas particulier, B_{E_C} croît avec le nombre et la valeur absolue des variables intermédiaires. Dans la pratique, on constate certains comportements pessimistes, voire même des dépassements de capacité (*overflow*). Nous verrons cependant avec les exemples qui suivent que cette borne B_{E_C} nous permet d'invalider certains résultats non corrigés et peut renseigner sur la sensibilité du calcul aux erreurs d'arrondi.

3 Applications caractéristiques

Les exemples suivants illustrent les principaux intérêts et limitations de l'application de la méthode CENA. Les calculs et les corrections sont effectués en simple précision IEEE-754 ; ce qui permet de comparer facilement correction et calculs en précision supérieure, soit ici en double précision. Nous rappelons qu'en arithmétique IEEE-754, la double précision est sensiblement plus précise que deux fois la simple précision (respectivement 53 et 24 bits de mantisse). L'environnement matériel des expérimentations présentées est le suivant : Sun Ultra5 workstation (SunOS 5.7, Solaris 1.3), FORTRAN 90 (WorkShop Compilers 5.0, FORTRAN 90 2.0).

3.1 Le résultat corrigé est le résultat exact

Considérons le produit scalaire X^TY avec $X = (1, 1, \cdots, 1)^T$ et $Y = (1, 2, \cdots, 2^n, -1, -2, \cdots, -(2^n))^T$. Ces vecteurs sont choisis pour générer une importante perte de précision lors du calcul de X^TY. Il n'y a pas de dépassement de capacité pour la dimension des vecteurs choisie. Le résultat exact vérifie $X^TY = 0$. Dans le tableau suivant, le résultat des calculs en simple et double précision de X^TY sont respectivement $\widehat{x}$ et $\widehat{x}_d$; la correction en simple précision de $\widehat{x}$ donne $\overline{x}$ et la borne associée B_{E_C}.

taille	$\widehat{x}$	$\overline{x}$	B_{E_C}	$\widehat{x}_d$
$2 \cdots 48$	0.000000E+00	0.000000E+00	0.000E+00	0.000000E+00
50	2.000000E+00	0.000000E+00	2.384E-05	0.000000E+00
52	4.000000E+00	0.000000E+00	4.863E-05	0.000000E+00
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
80	6.553600E+04	0.000000E+00	7.968E-01	0.000000E+00
82	1.310720E+05	0.000000E+00	1.593E+00	0.000000E+00
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
106	5.368709E+08	0.000000E+00	6.528E+03	0.000000E+00
108	1.073741E+09	0.000000E+00	1.305E+04	2.000000E+00
110	2.147483E+09	0.000000E+00	2.611E+04	4.000000E+00
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
218	3.868562E+25	0.000000E+00	4.703E+20	7.205759E+16
220	7.737125E+25	0.000000E+00	9.407E+20	1.441151E+17

Cet exemple permet d'illustrer plusieurs aspects très caractéristiques du calcul en précision finie et de la correction automatique.

- Pour un algorithme donné, augmenter la précision ne fait souvent que retarder l'effet des erreurs d'arrondi. Dans cet exemple, les calculs en simple et double précision souffrent d'une erreur globale qui croît comme $2^{n-n_{\mathbf{u}}}$ où $n > n_{\mathbf{u}}$, une borne qui dépend de la précision $\mathbf{u}$.
- La correction linéaire n'est pas équivalente à une augmentation de précision. En effet, le résultat corrigé $\overline{x}$ est ici égal au résultat exact, et ce pour toute dimension telle que le calcul initial de $\widehat{x}$ ne provoque pas de dépassement de capacité.
- Comparer les colonnes $\widehat{x}$ et B_{E_C} prouve que la valeur $\widehat{x}$ calculée en simple précision est fausse pour $n > 24$. En effet, on a $|\widehat{x}| > B_{E_C}$ tandis que le corollaire 1 assure $|x| \leq B_{E_C}$.
- Pourtant, le majorant B_{E_C} est d'autant plus pessimiste qu'ici $E_C = 0$. La borne B_{E_C} augmente comme la valeur absolue des plus grands termes rencontrés (ici comme une puissance de 2). Comme $\overline{x} = 0$, une telle valeur élevée pour B_{E_C} peut être un indicateur de la présence d'élimination catastrophique dans l'algorithme (soustraction de quantités proches en valeur absolue).
- Remarquons que ce calcul est équivalent à la somme de $2n + 2$ scalaires, les composantes de Y. En explicitant les dérivées partielles du terme correctif de la relation (5), il est facile de prouver que la correction automatique de ce calcul correspond à l'algorithme de sommation corrigé avec ajout final du terme correctif ou à la première itération de [19].

3.2 Le résultat corrigé est deux fois plus précis que le résultat initial

La « correction exacte » $\overline{x} = x$ de l'exemple précédent doit être considérée comme un cas très favorable. Rappelons qu'un résultat est optimal dés qu'il vérifie la majoration plus souple de la relation (3). Nous illustrons maintenant le comportement moyen de la méthode de correction linéaire par la résolution d'un système linéaire triangulaire inférieur $Tx = b$, de dimension n, par algorithme de descente $x_i = (b_i - \sum_{k=1}^{i-1} T_{ik}x_k)/T_{ii}$, pour $i = 1, \ldots, n$.

Nous construisons la matrice T et le second membre b de dimension n, à coefficients flottants tels que la précision de la solution calculée $\widehat{x}$ se dégrade (exponentiellement) avec n. Les composantes de la solution x ne sont pas des nombres flottants. Le système $Tx = b$ est explicité en Annexe. En choisissant T triangulaire inférieure, la perte de précision maximale dans $\widehat{x}$ vérifie $\max_{1 \leq i \leq n} |\widehat{x}_i - x_i|/|x_i| = |\widehat{x}_n - x_n|/|x_n|$.

Sur la figure 1, nous présentons la précision relative des solutions calculées en simple et double précisions, ainsi que celle de la solution simple précision corrigée et la borne relative associée $B_{E_C}/|x_n|$.

Les précisions des solutions non corrigées se dégradent de façon décalée mais similaire. Ce comportement est tout à fait en accord avec les majorations *a priori* classiques basées sur le conditionnement de T ; il illustre encore l'effet « retard » de l'augmentation de précision. La précision de la solution corrigée évolue différemment ; on distingue clairement trois régimes de correction.

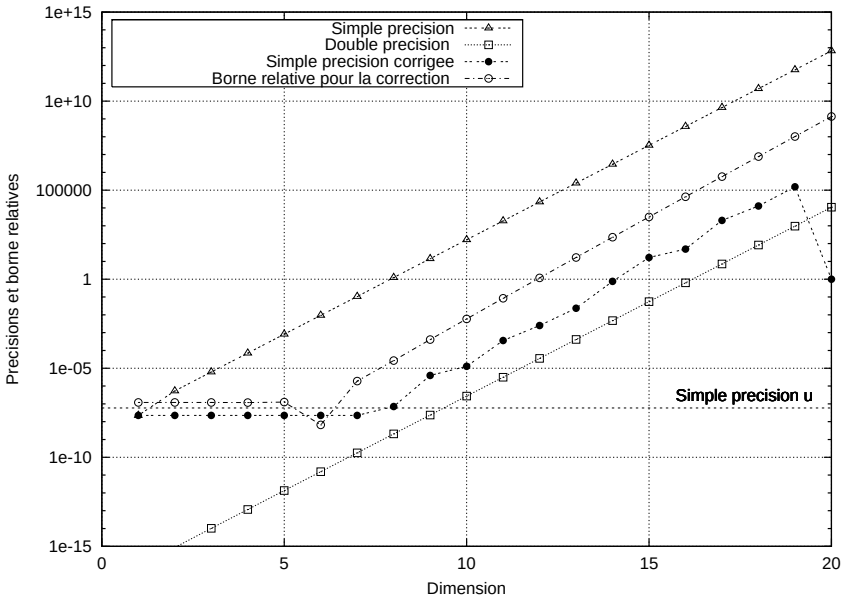

FIG. 1: Résolutions et correction de $Lx = b$ pour les dimensions 1 à 20.

1. La correction est optimale pour $n \leq 8$. Par construction la solution x n'appartient pas à $\mathbb{F}$. Retrouver une solution exacte est donc ici impossible, $fl(x)$ est la solution de meilleure précision. On vérifie que $\overline{x} \approx fl(x)$ pour ces valeurs de n.
2. La correction permet une précision de l'ordre[1] de celle fournie par la double précision pour $9 \leq n \leq 19$. C'est le comportement caractéristique d'une correction en précision $\mathbf{u}$ par ajout de terme correctif représenté par un flottant de précision $\mathbf{u}$ dans le cas suivant : le terme correctif est une bonne approximation de l'erreur effective et le résultat à corriger $\widehat{x}$ est d'une précision suffisamment dégradée pour qu'il soit d'un ordre de grandeur différent du résultat attendu. Remarquons en effet que l'erreur relative du résultat à corriger $\widehat{x}$ est supérieure à 1 pour $n \geq 9$.

   Illustrons schématiquement cette propriété en considérant par exemple la correction de $\widehat{x}_{11}$. Nous rappelons que la simple précision $\mathbf{u}_s$ est légèrement inférieure à $\mathbf{u} \approx 10^{-7}$. On a $\widehat{x}_{11} \approx -1000$, tandis que $x_{11} \approx 5$. L'erreur relative sur $\widehat{x}_{11}$ est de l'ordre de 10^3. La perte de précision du calcul en précision $\mathbf{u}$ s'interprète comme l'effet d'un facteur multiplicatif de l'ordre de 10^{11} sur la précision $\mathbf{u}$ initiale. Un terme correctif Δ de précision acceptable est de l'ordre de $\widehat{x}_{11}$. La précision absolue de la soustraction $\widehat{x}_{11} - \Delta$ est de l'ordre de $\mathbf{u}\Delta$, soit de l'ordre de 10^{-4} en précision $\mathbf{u}$, ce qui est aussi l'ordre de grandeur de la précision relative (pour x_{11}). La précision relative du résultat corrigé est donc de l'ordre de 10^{-4}, ce qui correspond bien à l'effet du facteur multiplicatif de l'ordre de 10^{11} sur une précision initiale de $\mathbf{u}^2 \approx 10^{-15}$.

   Ce développement se formalise en introduisant le conditionnement du problème considéré (le facteur multiplicatif) et la stabilité inverse de l'algorithme (la précision $\mathbf{u}$).
3. Le dernier régime de correction apparaît pour $n \geq 20$. La solution corrigée est $\overline{x}_n = 0$. Il s'agit là du cas extrême de la limitation précédemment vue. L'erreur absolue dans la solution calculée est telle que $|\widehat{x}|\mathbf{u} > |x|$. On a donc en précision $\mathbf{u}$, $fl(\Delta_L) = \widehat{x}$, et la soustraction de la correction finale (7) fournit alors $fl(\widehat{x} - \widehat{\Delta}_L) = 0$.

La borne B_{E_C} possède un comportement similaire. La borne relative $B_{E_C}/|x_n|$ majore l'erreur relative effective, de façon fine dans le premier régime de correction, puis de façon plus pessimiste ensuite. Remarquons que pour la dimension 6, la majoration $B_{E_C}/|x_6| < |\widehat{x} - x|/|x| < \mathbf{u}_s$ ne contredit pas la validation. Comme

[1] La double précision IEEE-754 est plus précise que deux fois la simple précision IEEE-754. Cette différence justifie le décalage qui apparaît sur la figure 1, entre les courbes de la simple précision corrigée et la double précision. Un exemple moins sensible à cette différence est proposé dans [15].

dans l'exemple précédent, la borne B_{E_C} permet aussi de prouver l'imprécision de la solution initialement calculée en simple précision.

Avec ces deux derniers régimes, nous exhibons les limites intrinsèques des méthodes de correction en précision finie $\mathbf{u}$ dans le cas où une seule correction finale est appliquée. La précision relative de la solution corrigée est limitée lors d'une soustraction en précision finie, par l'erreur absolue élevée dont est entaché l'opérande à corriger. Une étude théorique des limites d'existence d'un terme correctif est présentée dans [8].

3.3 La correction intermédiaire stabilise l'algorithme

L'exemple précédent suggère d'appliquer la correction automatique avant que le résultat calculé soit trop imprécis, soit donc avant la fin du calcul. Nous avons aussi déjà signalé l'intérêt de corriger des parties linéaires d'algorithmes qui ne le sont pas. Enfin, la stabilité des algorithmes dépend dans certains cas de la précision de calculs intermédiaires. C'est en particulier le cas de méthodes itératives de type Newton. Ces trois raisons justifient de pouvoir *corriger des variables intermédiaires* lors du calcul.

Il est bien connu que le calcul de racines de polynômes par l'itération de Newton est sensible aux erreurs d'arrondi au voisinage de racines multiples. Le calcul en précision finie peut s'avérer instable, ne pas s'achever correctement (division par zéro) et fournir des approximations décevantes de la racine cherchée [22]. Augmenter la précision de l'itération de Newton permet ici de limiter ces effets.

Lorsque le polynôme p et sa dérivée sont évalués par la méthode de Horner, nous pouvons utiliser la correction linéaire automatique et calculer l'itération de Newton corrigée suivante

$$x_{k+1} = x_k - \frac{\overline{p}(x_k)}{\overline{p'}(x_k)}, \quad k = 0, 1, \ldots . \tag{9}$$

Les valeurs $\overline{p}(x_k)$ et $\overline{p'}(x_k)$ sont obtenues en appliquant la correction linéaire automatique aux polynômes p et p' pour chaque itéré x_k.

Un autre aspect important du calcul en précision finie de l'itération (9) au voisinage de racines multiples est le choix du test d'arrêt. Nous ne détaillerons pas les différentes stratégies possibles. Comme nous disposons de valeurs corrigées $\overline{p}(x_k)$, nous choisissons d'arrêter l'itération (9) lorsque le résidu absolu vérifie $|\overline{p}(x_k)| < \sigma'$, où σ' est de l'ordre de σ, le plus petit nombre flottant strictement positif. Nous prenons $\sigma' = 1.0 \times 10^{-37}$, sachant que $\sigma \approx 1.2 \times 10^{-38}$ en simple précision IEEE-754.

Nous appliquons l'itération de Newton, sans et avec corrections, à la détermination de la racine de multiplicité 6 du polynôme $p(x) = (x-1)^6$, ce polynôme étant évalué sous sa forme développée selon la méthode de Horner. Les résultats suivants sont obtenus pour $x_0 = 2$; d'autres valeurs d'initialisation conduisent aux même conclusions. La figure 2 représente les itérés calculés sans correction en simple et double précisions, et l'itération (9) de la correction en simple précision. Les itérations non corrigées deviennent rapidement instables et s'arrêtent sur des approximations qui annulent fortuitement le résidu $p(x_k)$. On trouve par exemple $x_{100} = 1.1120\,8498$ en simple précision alors que certains itérés antérieurs sont des approximations plus précises de la racine cherchée. L'itération corrigée (9) apparaît stable et s'arrête pour $x_{29} = 1.0060\,0839$. La précision de cet approximant vérifie $|x_{29} - 1| \approx \mathbf{u}^{1/3}$; cette relation est en accord avec les estimations théoriques pour un calcul en précision $\mathbf{u}^2$. On retrouve ainsi le comportement déjà illustré par la résolution du système linéaire.

3.4 Application complémentaire des méthodes CESTAC et CENA

L'exemple précédent illustre l'intérêt de corriger automatiquement certaines variables intermédiaires. La difficulté réside donc dans l'identification de telles variables sensibles. A notre connaissance, il n'existe pas de méthode générale qui réponde à ce problème. Ce dernier exemple présente un cas où l'utilisation complémentaire de la méthode CESTAC et de la méthode CENA permet de localiser de telles variables puis d'améliorer suffisamment la précision de la solution cherchée pour identifier les causes de la perte de précision.

Nous supposons que le lecteur connaît les principes de la méthode CESTAC, et en particulier la notion de nombre de chiffres significatifs garanti par le logiciel CADNA (qui implante la méthode CESTAC) ainsi

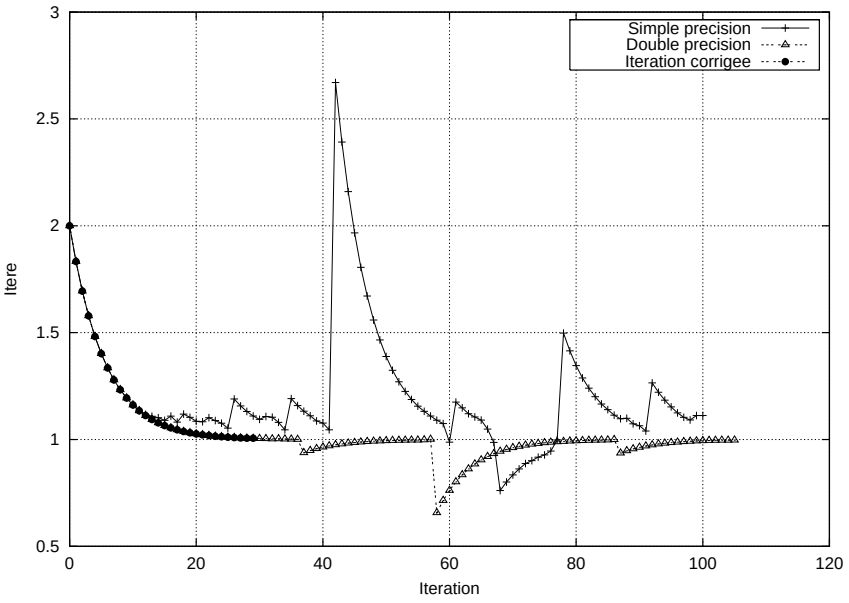

FIG. 2: Stabilisation de l'itération de Newton par correction.

que l'utilisation de la valeur spécifique @.0 , appelée zéro stochastique, dans les tests d'arrêt des méthodes itératives. Nous renvoyons le lecteur à [20], [2], ou [6] dans ce volume, pour le détail de cette méthode.

Considérons le calcul de la racine double $x^* = 3/7$ du polynôme $p(x) = 1.47x^3 + 1.19x^2 - 1.83x + 0.45$; son autre racine étant $y^* = -5/3$ [5]. Nous appliquons l'itération de Newton en simple et double précision IEEE-754 pour $x_0 = 0.5$. Ces itérations s'arrêtent respectivement pour $k = 16$ et $k_d = 25$ et donnent les approximants x et x_d qui annulent le résidu à la précision machine, c'est-à-dire $\widehat{p}(x) = \widehat{p}(x_d) = 0$, où $\widehat{p}$ est l'évaluation flottante de p dans la précision considérée. Les précisions relatives de x et x_d, respectivement $2.4E-5$ et $1.7E-9$, sont conformes aux résultats théoriques si on sait qu'il s'agit d'une racine double, ce qui n'est généralement pas le cas.

Le principe de l'utilisation conjointe des deux méthodes est assez naturel ici. Nous appliquons d'abord la méthode CESTAC pour arrêter l'itération de Newton dés que la précision de l'itéré x_k n'est plus améliorée. Dans notre cas, l'itération de Newton exécutée en arithmétique stochastique s'arrête pour $p(x_k) = @.0$ et $x_k = 0.4287$. Le logiciel CADNA garantit les 3 premiers chiffres de x_k.

Nous utilisons ensuite l'itération de Newton corrigée (9) pour améliorer la précision des itérés suivants. Le résultat garanti par CADNA suggère plusieurs valeurs d'initialisation x_0 telles que $x_{0-} \leq x_0 \leq x_{0+}$, avec les valeurs $x_{0-} = 0.4280\,0000$ et $x_{0+} = 0.4289\,9999$ pour la simple précision. Nous choisissons d'initialiser l'itération de Newton corrigée (9) avec les deux valeurs x_{0-} et x_{0+}. Le tableau suivant résume les calculs en simple précision.

$x(0)$	k	x_k	$\overline{p}(x_k)$	$\frac{\lvert x_k - x_{k-1}\rvert}{\lvert x_k\rvert}$	B_{E_C}
x_{0-}	7	0.4284 9594	3.70E-12	0.0	2.60E-14
x_{0+}	6	0.4286 4692	4.83E-12	0.0	9.19E-15

Ces deux calculs s'arrêtent l'un comme l'autre pour $|x_k - x_{k-1}|/|x_k| = 0$, ce qui assure la précision optimale de l'itération, mais ils convergent vers deux approximations $\overline{x}^*_- = 0.4284\,9594$ et $\overline{x}^*_+ = 0.4286\,4692$, qui diffèrent à partir du troisième chiffre. La borne B_{E_C} et la valeur du résidu corrigé $\overline{p}(x_k)$ prouvent qu'aucune de ces valeurs n'est un zéro du polynôme corrigé. Le choix d'autres valeurs d'initialisation entre x_{0-} et x_{0+} fournit des convergences similaires.

Cet exemple illustre un effet bien connu de la précision finie lorsque l'on s'intéresse au voisinage de racines multiples d'un polynôme à coefficients réels. La perturbation engendrée par le codage en précision finie de ses coefficients transforme la racine ici double en deux racines simples séparées x^*_- et x^*_+.

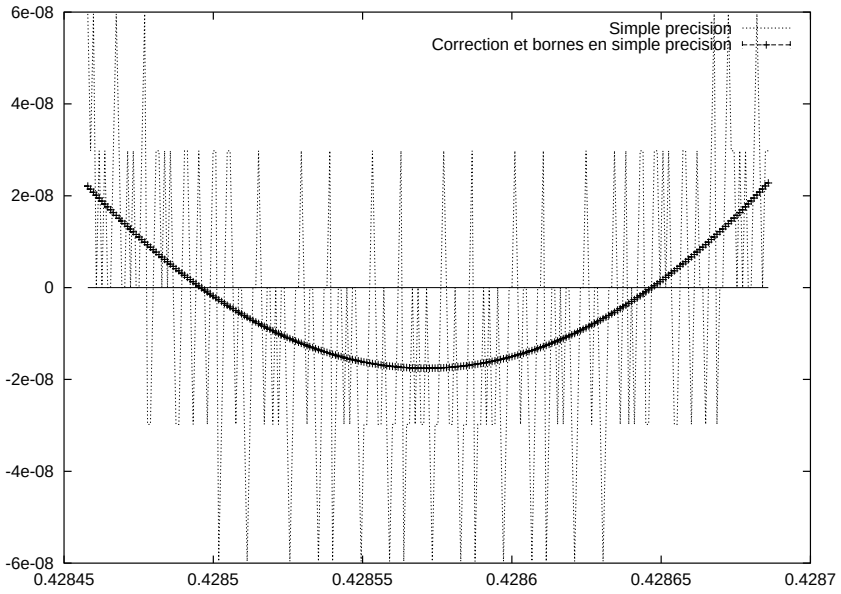

FIG. 3: Evaluations polynomiales avec et sans correction au voisinage d'une racine double (les bornes sont mesurées par les composantes verticales du +).

Nous vérifions que les valeurs $\overline{x}^*_-$ et $\overline{x}^*_+$ calculées par l'itération corrigée sont des approximations de précision optimale de x^*_- et x^*_+. Pour cela, nous utilisons le logiciel de calcul formel Maple pour localiser les racines du polynôme $\widehat{p}$. Nous définissons ce polynôme $\widehat{p}$ par $\widehat{p}(x) = \widehat{a}_3 x^3 + \widehat{a}_2 x^2 + \widehat{a}_1 x + \widehat{a}_0$, où $\widehat{a}_i = fl\,(a_i)$ est la valeur (rationnelle) exacte de la représentation binaire en simple précision du coefficient a_i de p $(0 \leq i \leq 3)$. En limitant la séparation des racines par suites de Sturm à des intervalles représentatifs de la simple précision, on obtient

$$I_- = [0.4284\,9593; 0.4284\,9594] \quad \text{et} \quad I_+ = [0.4286\,4691; 0.4286\,4692].$$

On vérifie que $\overline{x}^*_- \in I_-$ et $\overline{x}^*_+ \in I_+$. La figure 3 représente l'évaluation au voisinage de ces valeurs, du polynôme p en simple précision, ainsi que son évaluation corrigée $\overline{p}$ et les bornes associées B_{E_C}. L'expérience numérique est en accord avec le résultat symbolique. A ce niveau de précision, la correction permet d'identifier la séparation de la racine double.

4 Conclusion

Nous espérons vous avoir convaincu que la correction automatique est une voie intéressante pour améliorer la qualité numérique des logiciels. La correction est une alternative à la simple augmentation de précision, son automatisation la rend aussi facilement accessible. Il faudra maintenant apprécier les avantages (et les coûts) respectifs des deux approches.

Cette question est d'autant plus d'actualité que les travaux en cours de normalisation des BLAS, les briques de base du logiciel d'algèbre linéaire, intègrent pour la première fois des opérateurs qui peuvent utiliser une précision supérieure à la précision du calcul appelant [3]. Ces techniques ont permis récemment d'améliorer la précision *et* la performance de certains algorithmes scalaires ou parallèles [9]. Ces améliorations s'appuient sur l'utilisation des bibliothèques qui augmentent la précision dont nous avons parlé au début de cet article. Utiliser la correction automatique permettrait-elle des améliorations différentes ?

Par le biais de la méthode CENA, nous avons illustré les limitations intrinsèques à un processus de correction finale. Il nous semble avoir moins insisté sur les limitations propres à la méthode elle-même, comme par exemple celles engendrées par l'approximation d'ordre 1. Mais peut-être est-ce parce que la propagation des erreurs d'arrondi est, hors cas d'école, fortement linéaire ? Etait-ce ce dont voulait nous convaincre Linnainmaa, l'un des fondateurs des approches de majoration linéaires, en intitulant son dernier

article sur le sujet "Error Linearization as an Effective Tool for Experimental Analysis of the Numerical Stability of Algorithms" ? [17]

Nous devons maintenant distinguer deux niveaux de correction automatique. D'abord, les algorithmes « auto-correcteurs » qui corrigent automatiquement leurs calculs, comme par exemple les sommations simplement ou doublement compensées. Ensuite, des méthodes plus générales qui ont pour objectif de corriger des algorithmes, comme la méthode CENA que nous avons longuement discuté. La généralité de cette méthode altère l'efficacité de son application, à la fois en précision et en performance. Nous avons identifié des cas où la correction linéaire automatique générait des algorithmes « auto-correcteurs » existants. On en déduit aisément la bonne façon d'appliquer la méthode CENA : comme un outil interactif d'expérimentation numérique. Si on identifie des corrections intéressantes, les bases de l'algorithme « auto-correcteurs » sont trouvées. Bien sûr, cela ne sous-entend nullement que de telles méthodes automatiques rendent obsolètes les connaissances de bases du calcul en précision finie et l'intuition des problèmes numériques que possèdent l'ingénieur ou le chercheur dans ces domaines — au contraire !

5 Annexe

Les systèmes linéaires triangulaires $Tx = b$, de dimension 1 à 20, étudiés en section 3.2 sont définis à partir des matrice et vecteurs de dimension 20 suivants,

$$\begin{bmatrix} 100 & 0 & \cdots & \cdots & \cdots & \cdots & 0 \\ -24 & 1 & \ddots & & & & \vdots \\ 24 & -24 & 1 & \ddots & & & \vdots \\ -24 & 24 & \ddots & 1 & \ddots & & \vdots \\ & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 24 & \ddots & \ddots & \ddots & \ddots & 1 & 0 \\ -24 & 24 & & & 24 & -24 & 1 \end{bmatrix} \begin{bmatrix} 0.01 \\ -0.01 \\ 0.02 \\ \vdots \\ \vdots \\ \vdots \\ 1310.72 \\ -2621.44 \end{bmatrix} = \begin{bmatrix} 1 \\ -0.25 \\ 0.5 \\ \vdots \\ \vdots \\ \vdots \\ 32768 \\ -65536 \end{bmatrix}.$$

Le système de dimension n est le sous-système extrait du système de dimension 20 à partir de $T_{1,1}$. Les composants de T et b sont des flottants binaires, ce qui n'est pas le cas de x. On déduit les composantes de b (et de x) sachant que $b_{i+1}/b_i = -2$, $i \geq 2$ (et donc $x_{i+1}/x_i = -2$).

Références

[1] David H. Bailey. A Fortran-90 double-double library. Available at URL = `http://www.nersc.gov/~dhb/mpdist/mpdist.html`.

[2] Jean-Claude Bajard, Olivier Beaumont, Jean-Marie Chesneaux, Marc Daumas, Jocelyne Erhel, Dominique Michelucci, Jean-Michel Muller, Bernard Philippe, Nathalie Revol, Jean-Louis Roch, and Jean Vignes. *Quality of Computers Computations : Toward More Reliable Arithmetics ?* Masson, Paris, 1997. (In French).

[3] Document for the Basic Linear Algebra Subprograms (BLAS) standard. URL = `http://www.netlib.org/utk/papers/blast-forum.html`, February 2001.

[4] Keith Briggs. Doubledouble floating point arithmetic. Available at URL = `http://www-epidem.plantsci.cam.ac.uk/~kbriggs/doubledouble.html`, 1998.

[5] Jean-Marie Chesneaux. *Stochastic Arithmetic and CADNA software.* Habilitation à Diriger des Recherches Thesis, Université Pierre et Marie Curie, Paris, France, November 1995. (In French).

[6] Jean-Marie Chesneaux and Fabienne Jézéquel. Théorie et pratique de l'arithmétique stochastique discrète. *Calculateurs Parallèles, Réseaux, et Systèmes Répartis*, 2001. ce numéro, (In French).

[7] Germund Dahlquist and Åke Björck. *Numerical Methods.* Prentice-Hall, Englewood Cliffs, NJ, USA, 1974. Translated by Ned Anderson.

[8] Marc Daumas and Philippe Langlois. Additive symmetric : the non-negative case. *Theoret. Comput. Sci.*, 2001. (To appear).

[9] James W. Demmel. Recent progress in high accuracy and high performance linear algebra algorithms. SIAM Annual Meeting, May 1999. Available at URL = `http://www.cs.berkeley.edu/~demmel`.

[10] Andreas Griewank. *Evaluating derivatives. Principles and techniques of algorithmic differentiation.* Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2000.

[11] Yozo Hida, Xiaoye S. Li, and David H. Bailey. Quad-double arithmetic : Algorithms, implementation, and application. Technical Report LBNL-46996, Department of Computer Science, University of California, Berkeley, CA, USA, oct 2000.

[12] Nicholas J. Higham. *Accuracy and Stability of Numerical Algorithms.* Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1996.

[13] *IEEE Standard for Binary Floating-Point Arithmetic, ANSI/IEEE Standard 754-1985.* Institute of Electrical and Electronics Engineers, New York, 1985. Reprinted in SIGPLAN Notices, 22(2) :9–25, 1987.

[14] William Kahan. Further remarks on reducing truncation errors. *Communications of the ACM*, 8(1) :40, 1965.

[15] Philippe Langlois. Automatic linear correction of rounding errors. *BIT*, 41(3), September 2001. (To appear).

[16] Philippe Langlois and Fabrice Nativel. When automatic linear correction of rounding errors is exact. *C.R. Acad. Sci. Paris, Série 1*, 328 :543–548, 1999. Erratum in 328 :829, 1999.

[17] Seppo Linnainmaa. Error linearization as an effective tool for experimental analysis of the numerical stability of algorithms. *BIT*, 23 :346–359, 1983.

[18] Daniel D. McCracken and William S. Dorn. *Numerical Methods and Fortran Programming : With Applications in Science and Engineering.* Wiley, New York, 1964.

[19] Michèle Pichat. Correction d'une somme en arithmétique à virgule flottante. *Numer. Math.*, 19 :400–406, 1972.

[20] Michèle Pichat and Jean Vignes. *Ingénierie du contrôle de la précision des calculs sur ordinateur.* Technip, Paris, 1993. (In French).

[21] Douglas M. Priest. *On Properties of Floating Point Arithmetics : Numerical Stability and the Cost of Accurate Computations.* PhD thesis, Mathematics Department, University of California, Berkeley, CA, USA, November 1992. URL = `ftp://ftp.icsi.berkeley.edu/pub/theory/priest-thesis.ps.Z`.

[22] Louis B. Rall. Convergence of the Newton process to multiple solutions. *Numer. Math.*, 9 :25–37, 1966.

[23] T. G. Robertazzi and S. C. Schwartz. Best "ordering" for floating-point addition. *ACM Trans. Math. Software*, 14(1) :101–110, 1988.

[24] J. Stoer and R. Bulirsch. *Introduction to Numerical Analysis.* Springer-Verlag, New York, second edition, 1993.

[25] James H. Wilkinson. Error analysis revisited. *IMA Bulletin*, 22(11/12) :192–200, 1986.

[26] Paul Zimmermann. Arithmétique en précision arbitraire. *Calculateurs Parallèles, Réseaux, et Systèmes Répartis*, 2001. ce numéro, (In French).

Unit´e de recherche INRIA Lorraine, Technopôle de Nancy-Brabois, Campus scientifique,
615 rue du Jardin Botanique, BP 101, 54600 VILLERS LÈS NANCY
Unit´e de recherche INRIA Rennes, Irisa, Campus universitaire de Beaulieu, 35042 RENNES Cedex
Unit´e de recherche INRIA Rhône-Alpes, 655, avenue de l'Europe, 38330 MONTBONNOT ST MARTIN
Unit´e de recherche INRIA Rocquencourt, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex
Unit´e de recherche INRIA Sophia-Antipolis, 2004 route des Lucioles, BP 93, 06902 SOPHIA-ANTIPOLIS Cedex

Éditeur
INRIA, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex (France)
http://www.inria.fr
ISSN 0249-6399